Sprawozdanie nr 2
Praktyka i teoria szeregowania zadań

Data: 03.12.2020
Autor: Damian Tabaczyński
Indeks: inf136819
Grupa dziekańska: L6 (przeniesienie z L5)
Zajęcia: czwartek 11:45

1. Opis problemu

Problemem rozważanym w niniejszym sprawozdaniu jest problem szeregowania **n** zadań na pięciu jednorodnych maszynach równoległych, opisanych współczynnikiem prędkości $\mathbf{b_k}$. Głównym celem jest minimalizacja średniego czasu przepływu **F** wyrażonego wzorem**:**

$$F = \frac{1}{n} \sum_{j=1}^{n} F_j$$

Zagadnienie to można przedstawić również w notacji trójpolowej:

Q5 | r_j | F

Ważną cechą tego konkretnego problemu jest niepodzielność zadań, wykonywanie zadań bez możliwości ich przerwania. Niezbędnym założeniem jest by każda maszyna wykonywała tylko jedno zadanie i jedno zadanie było wykonywane przez jedną maszynę w tej samej jednostce czasu. Dodatkowo żadne zadanie nie może rozpocząć się przed swoim momentem gotowości $\mathbf{r_j <= C_j - p_j}$.

Zgodnie z nomenklaturą podaną w zadaniu oraz przedstawioną na wykładzie:

- każde zadanie $\mathbf{J_j}$ opisane jest czasem trwania $\mathbf{p_j}$ oraz momentem (czasem) gotowości $\mathbf{r_j}$
- $\mathbf{C_j}$ oznacza moment zakończenia wykonywania zadania $\mathbf{J_j}$ w uszeregowaniu
- $\mathbf{b_k}$ oznacza współczynnik prędkości - wskazuje ile razy maszyna $\mathbf{M_k}$ jest wolniejsza od najszybszej maszyny w systemie
- $\mathbf{F_j}$ oznacza czas przepływu czyli różnice czasu zakończenia $\mathbf{C_j}$ i czasu gotowości zadania $\mathbf{r_j}$

W ramach rozwiązania należy opracować i przedstawić generator instancji oraz algorytm listowy rozwiązujący przedstawiony problem, który nie przekracza złożoności obliczeniowej **$O(n^2 \log n)$.**

2. Opis generatora instancji

W całym algorytmie generowania instancji zmienna **max_length** oznacza maksymalną wartość $\mathbf{p_j}$ zadania i wynosi **20**.

Generator instancji rozpoczyna swoje działanie od wygenerowania pięciu współczynników prędkości $\mathbf{b_k}$ dla każdej z maszyn osobno. Jednej z pseudolosowo wybranych maszyn generator przypisuje współczynnik równy 1 (wymóg by logika zadania była spójna). Pozostałym czterem maszynom generator wybiera pseudolosową liczbę z dyskretnego przedziału (1,25; 1,5 ; 1,75 ; … ; 4,75 ; 5,0).

Kolejnym etapem jest wygenerowanie pięciu zadań, których czasy gotowości $\mathbf{r_j}$ wynoszą **0**. Czasy trwania $\mathbf{p_j}$ tych zadań są liczbą pseudolosową z zakresu **<1, max_length>.** W domyśle zadania te powinny być jako pierwsze uszeregowane przez potencjalny algorytm. Po ich wygenerowaniu zmienna **time** (niezbędna w następnym etapie, początkowa wartość 0) jest inkrementowana o pseudolosowo wybraną wartość $\mathbf{p_j}$ ze stworzonych zadań.

W następnym kroku zadania tworzone są **“piątkami”** tzn. iteracyjnie w pętli tworzone jest pięć nowych zadań aż do osiągnięcia docelowej liczby **n** zadań. Każdemu takiemu zdaniu jest przypisywany czas trwania $\mathbf{p_j}$ jako liczba pseudolosowa z zakresu **<1, max_length>.** Trzem wybranym pseudolosowo zadaniom z każdej **“piątki”** przydzielana jest wartość momentu gotowości $\mathbf{r_j}$ równa zmiennej **time** (czyli czasu po jakim wykona się zadanie z poprzedniej piątki jeśli byłoby wykonywane na maszynie o $\mathbf{b_k}$=1). Pozostałym dwóm zadaniom z **“piątki”** przydzielana jest pseudolosowo wartość $\mathbf{r_j}$ z zakresu **<time+20, time+40> czyli <time + max_length, time + 2*max_length>**. Po wybraniu **“piątki”** w każdej z iteracji następuje inkrementacja zmiennej **time** o jedną z wartości $\mathbf{p_j}$ wygenerowanego zadania w tej iteracji. Wartość ta jest pseudolosowo wybierane z jednego z zadań tzw. **“piątki”.**

Na koniec indeks każdego zadania jest przydzielany pseudolosowo, więc prawdopodobieństwo sytuacji w której zadania znajdujące się w pliku instancji byłyby w kolejności generowania jest bliska zeru.

Dzięki takiemu sposobowi generowaniu danych potencjalnemu algorytmowi będzie ciężej uszeregować odpowiednie zadanie na każdą maszynę (szczególnie na maszynie o współczynniku $\mathbf{b_k}$ = 1), aby nie doszło do przestojów.

3. Opis algorytmu

Algorytm bazuje na rozwiązaniu listowym. Na początku sortowana jest lista wszystkich zadań rosnąco według momentu gotowości $\mathbf{r_j}$ każdego zadania. W przypadku gdy momenty te są równe to o kolejności decyduje czas trwania $\mathbf{p_j}$ - jeśli mniejszy czas trwania to indeks również mniejszy (sortowanie rosnąco).

W przypadku sortowania został użytu algorytm Quicksort posiadający średnią złożonością **O(n log n)**, a najgorszą $\mathbf{O(n^2)}$. Po wstępnym ułożeniu zadań następuje właściwe uszeregowanie:

Założenia:

- Lista dostępnych zadań - lista posortowanych uprzednio zadań.
- Każda maszyna posiada swój własny licznik czasu, który jest aktualizowany po wykonaniu zadania.

Lista kroków:

1. Dopóki istnieje choć jedno nieuszeregowane zadanie (lista dostępnych zadań jest większa od 0) to kontynuuj, w przeciwnym przypadku zakończ działanie algorytmu.

2. Wybierz najmniejszy czas spośród wszystkich liczników czasu maszyn i przypisz do zmiennej **min_time**.

3. Utwórz listę gotowych zadań. Lista ta zawiera zadania, których moment gotowości $\mathbf{r_j}$ jest mniejszy lub równy zmiennej **min_time** wyznaczonej w poprzednim punkcie.

4. Jeśli **lista gotowych zadań** jest pusta to wybierz zadanie znajdujące się na początku **listy dostępnych zadań**. W przeciwnym przypadku wybierz zadanie o najkrótszym czasie trwania $\mathbf{p_j}$ (czyli minimum po **liście gotowych zadań**). Usuń wybrane zadanie z **listy dostępnych zadań**.

5. Posortuj listę maszyn rosnąco po wartości licznika czasu. W przypadku gdy liczniki te są równe to o kolejności decyduje współczynnik $\mathbf{b_k}$ - jeśli mniejszy współczynnik to indeks również mniejszy (sortowanie rosnąco).

6. Wybierz maszynę o najmniejszym indeksie spośród posortowanej listy maszyn. Na tej maszynie wykonaj zadanie wybrane w **punkcie 4.**

7. Powróć do **punktu 1.**

Algorytm ten faworyzuje wykonywanie krótkich gotowych zadań na maszynach o najmniejszym liczniku czasu (tzn. na maszynach, które nie są zajęte w porównaniu do innych w danej chwili | są gotowe do działania) by zminimalizować ilość

przestojów. Dodatkowo uprzywilejowane są również maszyny mające niższy współczynnik $\mathbf{b_k}$.

Warto zauważyć, iż jeśli nie istnieją zadania gotowe przy danej kombinacji liczników czasu maszyn to algorytm wybierze pierwsze zadanie z posortowanej listy zadań - oznacza to przestój w szeregowaniu.

Złożoność początkowego sortowania Quicksortem listy zadań wynosi **O(n log n)**. Następnie algorytm wykonuje w pętli **n** razy:

- operacje **min** po maszynach czyli **O(5)**
- tworzenie listy gotowych zadań - w najgorszym przypadku wszystkie zadania mogą być gotowe - **O(n)**
- jeśli lista nie jest pusta to **min** - w najgorszym przypadku gdy wszystkie zadania są gotowe - **O(n)**
- sortowanie maszyn - **O(5 log5)**

Podsumowując złożoność algorytmu wynosi **O(n log n + 5n + 2n^2 + 5n log5)**, co sprowadza się do złożoności **O(n^2)**.

4. Wyniki algorytmu i obserwacje

n	Wartość kryterium dla sztucznego pliku wynikowego **A**	Wartość kryterium uzyskana przez własny algorytm **B**	Względna różnica w [%] **C = (A-B)/A *100**	Czas działania algorytmu [µs]
50	194,32	56,59	70,88	941097
100	397,62	99,44	74,99	965430
150	399,51	77,73	80,54	949033
200	714,37	190,47	73,34	1099104
250	948,07	173,03	81,75	930915
300	856,55	142,18	83,4	970392
350	1124,98	176,16	84,34	962899
400	1545,76	352,7	77,18	980861
450	1483,99	291,38	80,37	978646
500	2222,24	474,9	78,63	1132107

Średnia wartość **C** dla moich instancji: **78,54%**
Średnia wartość **C** dla całej biblioteki: **67,89%**

Jak widać w przedstawionych danych algorytm osiąga bardzo duże względne różnice dla wygenerowanych instancji własnych. Są one średnio na poziomie **78,54%**. Warto zauważyć, że równie dobrze poradził sobie na instancjach całej biblioteki o czym świadczy średnia różnica **C** na poziomie **67,89%**. Algorytm osiąga ponadprzeciętne osiągi dla całej biblioteki instancji oraz w większości przypadków najlepszy wynik kryterium, co można ewidentnie zaobserwować w pliku z zestawieniami wyników w arkuszu “kryterium”.